AIPS MEMO NO. 41

I/O SPEEDS ON A VAX 11/780

R. G. Noble.
NRAO and NRAL*
5th December, 1985.

Introduction

This document describes the results of various timing measurements on VAX I/O speeds. They were carried out on a moderately loaded VAX 11/780.

Unless otherwise stated, all of the times below refer to writing or reading a 10,000 block file, where one block is 512 bytes. The file was preallocated before writing, and was contiguous-best-try. The record size on the file was 8192 bytes, which corresponds to the default VMS Record Managment Services (RMS) multi-block count; i.e., RMS actually writes 16 512-byte blocks at a time.

When comparing the elapsed times in the following, it should be remembered that these are significantly affected by machine loading. Thus, the times should only be taken as a guide. For example, the elapsed times for the use of QIOs and block mode I/O imply that block mode is faster. This is unlikely to be the case since block mode calls QIOs! The best that can be said is that the elapsed times for these two means of I/O are practically the same.

Using QIOs

The QIO is the basic I/O mechanism in VMS and so it might be expected that it will be the fastest. The disadvantage of using QIOs is that RMS can *only* be used to open the file: its use thereafter is precluded. This means that the things that RMS makes easy to do have to be done using QIO calls.

The I/O operations were done transferring 8192 bytes per QIO.

Writing Elapsed time = 27 secs. Cpu time = 1.5 secs.

Reading Elapsed time = 39 secs. Cpu time = 1.7 secs.

* *Visiting Scientist from Nuffield Radio Astronomy Lab., Jodrell Bank, Macclesfield, Cheshire SK11 9DL, ENGLAND*

Using block mode I/O

RMS block mode I/O (WRITE and READ) is an intermediate case between using QIOs and using the full RMS facilities. Block mode does not preclude using other RMS facilities as the use of QIOs does. It can also be used over DECnet.

The I/O operations were done transfering 8192 bytes at a time.

Writing Elapsed time = 25 secs. Cpu time = 2.0 secs.

Reading Elapsed time = 25 secs. Cpu time = 2.2 secs.

Block mode was also tried via DECnet. This was a DECnet transfer to the same machine, so no link overhead was incurred.

Writing Elapsed time = 1m 27 secs. Cpu time = 28.8 secs.

Reading Elapsed time = 1m 46 secs. Cpu time = 28.6 secs.

Both of the above times used *sequential only* access to the file (the SQO option in the FAB FOP field). If this cannot be done, the speed is significantly slower: for example

Writing Elapsed time = 7m 42 secs. Cpu time = 1 m 13.5 secs.

The sequential only transfer did 1250 I/O operations; the non-sequential one did 20,000 I/O operations.

There was no detectable overhead in sending the file over Ethernet.

Using record mode I/O

Record mode I/O is the normal means that RMS uses to do I/O (GET and PUT). This uses all of the RMS internal buffering.

The I/O operations were done transferring 8192 bytes at a time.

Writing Elapsed time = 30 secs. Cpu time = 7.3 secs.

Reading Elapsed time = 34 secs. Cpu time = 6.6 secs.

Note that the elapsed time is much the same as when using QIOs, but the cpu time is a lot more. This reflects the increased overhead of the RMS record handling that does not show up in the elapsed times because it can partly go on simultaneously with the actual I/O transfers.

If records do not cross block boundaries then it is possible to do *locate* mode input. In normal (*move*) mode input RMS copies the contents of its internal buffers to the buffer requested by the calling program; in locate mode input RMS simply returns a pointer to

the location of the required record in its own buffers. This saves a copy operation. The times for a *move* mode operation (writing 10,000 512 bytes records) are:

Writing Elapsed time = 28 secs. Cpu time = 8.7 secs.

Reading Elapsed time = 27 secs. Cpu time = 9.6 secs.

In comparison the times for *locate* mode are:

Reading Elapsed time = 26 secs. Cpu time = 5.7 secs.

The difference in cpu time (about 4 seconds) represents the time taken to copy 10000 times 512 bytes.

Using paging

The VAX paging mechanism can be used to effectively write and read files. In this case the file becomes part of the virtual memory of a program. The times to "write" and "read" 10,000 512-byte pages are:

Writing Elapsed time = 48 secs. Cpu time = 5.2 secs.

Reading Elapsed time = 30 secs. Cpu time = 4.8 secs.

These times include an overhead because there must be addtional code to write to and read from each page to make sure that it is actually transferred. However, it is apparent that there is no significant advantage in using paging as opposed to QIOs or move mode I/O.

FORTRAN

As a comparison, the times taken by a FORTRAN code program to do the same I/O are given:

The I/O operations were done transferring 8192 bytes at a time.

Writing Elapsed time = 37 secs. Cpu time = 7.8 secs.

Reading Elapsed time = 49 secs. Cpu time = 7.4 secs.

It is interesting to see the additional overhead incurred if the file is *not* preallocated:

Writing Elapsed time = 1m 30 secs. Cpu time = 12.57 secs.

The I/O involved with this is 1025 direct transfers and 257 buffered ones, as opposed

to 625 direct transfers only if the file was allocated in advance.

Conclusions

To obtain the fastest I/O with minimum impact on the machine then QIOs are the preferred mechanism. However, the additional overhead of using RMS block mode I/O is very small, and gives the advantage of access to all other RMS functions (and DECnet). If elapsed time is the only criterion, then there is no significant difference between any of the different means of I/O.